Solving Footstep Planning as a Feasibility Problem using L1-norm Minimization (Extended Version)

Daeun Song[1], Pierre Fernbach[2], Thomas Flayols[2], Andrea Del Prete[3], Nicolas Mansard[2]
Steve Tonneau[4], and Young J. Kim[1]

***Abstract*—One challenge of legged locomotion on uneven terrains is to deal with both the *discrete problem* of selecting a contact surface for each footstep and the *continuous problem* of placing each footstep on the selected surface. Consequently, footstep planning can be addressed with a Mixed Integer Program (MIP), an elegant but computationally demanding method, which can make it unsuitable for online planning. We reformulate the MIP into a cardinality problem, then approximate it as a computationally efficient $\ell 1$-norm minimisation, called SL1M. Moreover, we improve the performance and convergence of SL1M by combining it with a sampling-based root trajectory planner to prune irrelevant surface candidates.**

Our tests on the humanoid Talos in four representative scenarios show that SL1M always converges faster than MIP. For scenarios when the combinatorial complexity is small (< 10 surfaces per step), SL1M converges at least two times faster than MIP with no need for pruning. In more complex cases, SL1M converges up to 100 times faster than MIP with the help of pruning. Moreover, pruning can also improve the MIP computation time. The versatility of the framework is shown with additional tests on the quadruped robot ANYmal.

***Index Terms*—Humanoid and Bipedal Locomotion, Legged Robots, Motion and Path Planning**

I. Introduction

FOOTSTEP planning consists of computing a sequence of footstep positions on which a legged robot should step to reach a desired goal position. It is thus a crucial problem for legged locomotion.

Footstep planning can be characterised by its combinatorial aspect. The parts of the environment where the footsteps can be placed (contact surfaces) are often disjoint. Thus, a discrete choice of a surface must be made. This choice has an impact on all future footstep locations, as they are constrained relative to each other by non-linear kino-dynamic constraints. This means that the discrete choices of contact surfaces must be considered simultaneously for all footsteps.

Several relaxations have been proposed to practically address the problem of footstep planning. The model size can be reduced by approximating the robot with a point mass model [1], [2], [3] and expressing the constraints as simplified functions of the Center of Mass (COM) and contact positions only [4], [5], [6]. The price for these approximations is paid with the loss of completeness [7], [8] and/or the guarantees of convergence [9], [10], [11]. Which part of the problem can be approximated is thus a fundamental question closely related to the issue of modelling the combinatorics. Discrete approaches embrace the discrete nature of the problem [12], [13], [14], [15] , while continuous, optimisation-based approaches have to design strategies to handle the discrete variables. In [16], a continuation method is employed to first solve a simplified version of the problem. Then the solution is used as a starting point for subsequent problems of increasing difficulty until the original problem is solved. Another option is to implicitly represent the discrete variables with continuous ones and to approximate the environment as a continuous function [9]. Alternatively, complementarity constraints can be used to enforce contact decisions, resulting in a computationally demanding, non-smooth optimisation problem [17], [18].

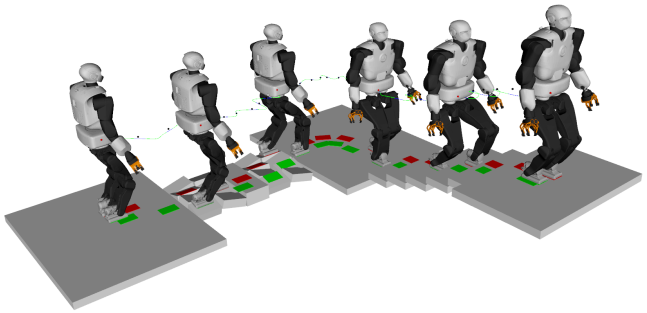

Fig. 1. A contact sequence generated by our convex-relaxed approach with domain-specific information. Talos robot is executing planned locomotion.

Though often conflicted in the literature, discrete and continuous approaches can be unified in the Mixed-Integer Programming (MIP) formalism [19], [20], [21], [22], [23]. Internally, MIP solvers first solve an approximated problem, in which they relax the discrete (integer) variables into continuous ones. In the best-case scenario, the relaxed variables converge to integer values, and the original problem is solved. Otherwise, the combinatorics have to be addressed with branch-and-bound strategies. In the worst case, this requires solving an optimisation problem for every possible combination of values for the integer variables. Fortunately, this is rarely needed in practice as efficient approaches exist to solve only a fraction of the combinations [24]. MIP solvers can efficiently handle

Corresponding author: Young J. Kim.
[1]D. Song and Y. J. Kim are with the Department of Computer Science and Engineering, Ewha Womans University, Seoul 03760, South Korea daeunsong@ewhain.net, kimy@ewha.ac.kr
[2]P. Fernbach, T. Flayols and N. Mansard are with the CNRS, LAAS, Université de Toulouse, Toulouse 31400, France pfernbach@laas.fr, tflayols@laas.fr, nmansard@laas.fr
[3]A. Del Prete is with the Department of Industrial Engineering, University of Trento, Via Sommarive 9, Trento 30123, Italy andrea.delprete@unitn.it
[4]S. Tonneau is with the University of Edinburgh, IPAB, Edinburgh EH8 9YL, U.K. stonneau@ed.ac.uk

convex problems, but struggle to address non-linear problems efficiently [25], which is why MIP-based approaches such as [19] work with linearised formulations.

In the best case, solving MIP is computationally equivalent to solving a continuous formulation of the problem, where no specific scheme is required to address the combinatorics. From this perspective, our research question is: *is it always possible to end up with a best-case scenario for the MIP in the context of footstep planning?* A positive answer would imply that the problem can be addressed effectively by off-the-shelf optimisation solvers, resulting in simpler and faster formulations of the contact planning problem. In this paper, we empirically show a positive result for a diverse range of instances of the contact planning problem, featuring stairs, rubbles, and narrow passages.

This paper is an extension of our earlier work [26], where we proposed a convex relaxation of the MIP approach for footstep planning with an $\ell 1$-norm minimisation, SL1M. In this preliminary work [26], SL1M has been shown to converge to integer solutions for "simple" problems, but fails when the combinatorial complexity becomes too high (as defined in Section VI). This issue is the focus of this paper.

A. Main Contributions

In this paper, we considerably improve our earlier work on SL1M by reducing the complexity of footstep planning problems. This is achieved by automatically removing non-relevant integer combinations using a low-dimensional path planner that computes a trajectory for the root of the robot [27]. We verified that improved SL1M always converges to an integer solution in our test scenarios after the pruning. We experimentally validate our hypothesis that we can reach a best-case scenario for the MIP in footstep planning by pruning the irrelevant combinations.

Thus, our main contribution is an LP relaxation of the MIP formulation, SL1M, experimentally shown to outperform commercial MIP solvers. Our second contribution is a demonstration that using a trajectory planner also improves the computational performance of MIP problems, especially for complex scenarios. These findings are demonstrated in several scenarios involving biped and quadruped robots navigating across uneven terrains with a pre-established gait.

II. Rationale

We study the combinatorial aspect of contact planning that results from environmental constraints. For instance, when climbing a staircase, we must decide which steps (or contact surfaces) our foot must land on. Planning the next n steps considering m possible contact surfaces for each step involves solving m^n optimisation problems in the worst case under the MIP formalism. We want to reformulate this problem so that we can find a solution by solving a single optimisation problem (or a few).

To achieve this objective, we first recall how to write the footstep planning problem as a MIP. Then we follow two paths of action. First, we automatically reduce the number of possible combinations. In our staircase example, if the robot starts at the bottom of the stairs, it is useless to consider the tenth step surface when planning the very first footstep. In general, any contact surface that is not in the reachable space of the foot can be discarded from the set of candidates. We automate this pruning process with a low-dimensional sampling-based trajectory planner [28]. Second, we verify that solving a feasibility problem requires fewer iterations than solving a minimisation problem. More interestingly, after recalling that the feasibility problem can be relaxed into an $\ell 1$-norm minimisation problem, we empirically show that the relaxed problem *always* converges to a feasible solution when the combinatorics is first reduced using our trajectory.

Conversely, our results show that currently, our continuous optimisation-based formulation does not work well if an objective has to be minimised simultaneously. However, this issue can be alleviated: once the contact surfaces have been selected by the $\ell 1$-norm relaxation of the feasibility problem, we can solve a second problem, which minimises a cost function but without modifying the contact surfaces selected by the first problem. This second problem boils down to a simple convex Quadratic Program (QP) [19], [26].

TABLE I
Table of Notation

Notation	Description
n	the number of footsteps in the planning
m	the number of contact surfaces in the environment
$\mathcal{S}$	union of potential contact surfaces available
$\mathcal{S}^j \subset \mathcal{S}$	the j-th contact surface
$\mathcal{S}_i \subset \mathcal{S}$	the contact surfaces considered in the i-th footstep
$\mathbf{p}_i \in \mathbb{R}^3$	i-th footstep position
$\mathbf{a}_i^j \in \{0,1\}$	integer slack variable for j-th surface in i-th footstep
$\boldsymbol{\alpha}_i^j \in \mathbb{R}^+$	positive real slack variable for j-th surface in i-th footstep
$\boldsymbol{\beta}_i^j \in \mathbb{R}$	real slack variables for j-th surface in i-th footstep
$\mathbf{R}_i \in SE(3)$	root position / orientation when creating footstep i
$\mathbf{card}(\mathbf{a})$	number of non-zero entries in vector $\mathbf{a}$
$\mathcal{I}, \mathcal{G}$	initial / goal constraint sets
$\mathcal{F}$	feasibility constraint set

III. Definitions and Notation

Table I defines the notation used throughout the paper. Unless specified, we use subscript for indicating the i-th footstep and superscript for indicating the j-th contact surface.

We now more specifically define the environment. The environment is represented as a union of m disjoint sets $\mathcal{S} = \bigcup_{j=1}^{m} \mathcal{S}^j$ as shown in Fig. 2. Each set $\mathcal{S}^j$ represents a convex contact surface, that is a polygon embedded in a 3D plane and bounded by a set of half-spaces:

$$\mathcal{S}^j := \{\mathbf{p} \in \mathbb{R}^3 | \mathbf{p}^T \mathbf{n}^j = e^j, \mathbf{S}^j \mathbf{p} \leq \mathbf{s}^j\}. \tag{1}$$

The equality defines the plane containing the contact surface, given by the normal $\mathbf{n}^j \in \mathbb{R}^3$ and $e^j \in \mathbb{R}$. $\mathbf{S}^j \in \mathbb{R}^{h \times 3}$ and $\mathbf{s}^j \in \mathbb{R}^h$ are respectively a constant matrix and a vector defining the h half-spaces that bound the quasi-flat[1] [29] surface. $\mathcal{S}_i \subset \mathcal{S}$ is the subset of contact surfaces that are considered for the i-th footstep, which we call the candidate surfaces.

[1]A surface is quasi-flat if the associated friction cone contains the gravity direction.

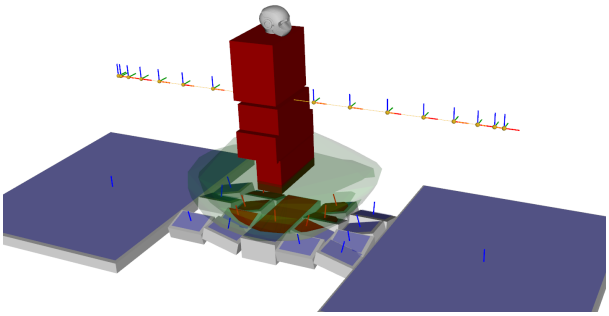

Fig. 2. The environment is a set of contact surfaces (blue and red). The reachable workspace of each foot is shown in green. The intersection between the reachable workspace of the left foot and the environment (red) defines potential contact surfaces. The normals of the contact surfaces (arrows) constrain the orientation of the foot around the x- and y- axes.

IV. Footstep Planning as a MIP

The section recalls casting the footstep planning problem as a MIP problem, adapted from [26] and originally introduced in [19]. We write the footstep planning problem as:

$$\begin{aligned} \textbf{find} \quad & \mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{3\times n} \\ \textbf{min} \quad & l(\mathbf{P}) \\ \textbf{s.t.} \quad & \mathbf{P} \in \mathcal{I} \cap \mathcal{G} \cap \mathcal{F} \\ & \mathbf{p}_i \in \mathcal{S} \quad \forall i, 1 \leq i \leq n\,. \end{aligned} \tag{2}$$

We want to find a user-defined number (for now) n of footstep positions $\mathbf{p}_i$. Optionally we may want to minimise an objective $l(\mathbf{P})$. $\mathbf{P}$ must satisfy initial and goal conditions, where simple conditions can be given by the sets $\mathcal{I} : \{\mathbf{P}, \mathbf{p}_1 = \mathbf{p}_\mathcal{I}\}$ and $\mathcal{G} : \{\mathbf{P}, \mathbf{p}_n = \mathbf{p}_\mathcal{G}\}$, with user-defined constants $\mathbf{p}_\mathcal{I}$ and $\mathbf{p}_\mathcal{G}$. $\mathcal{F}$ denotes the set of kinematic and dynamic feasibility constraints that guarantee the robot to follow the footstep plan without falling or violating joint limits, detailed in Appendix A. We constrain the positions of $\mathbf{P}$ to lie on a surface in $\mathcal{S}$.

With the surface constraints (1), the constraint $\mathbf{p}_i \in \mathcal{S}$ can then be formulated with the use of slack variables:

$$\begin{aligned} \textbf{find} \quad & \mathbf{p}_i \in \mathbb{R}^3 \\ & \mathbf{a}_i = [a_i^1, \cdots, a_i^m] \in \{0,1\}^m \\ & \boldsymbol{\beta}_i = [\beta_i^1, \cdots, \beta_i^m] \in \mathbb{R}^m \\ \textbf{s.t.} \quad & \mathbf{card}(\mathbf{a}_i) = m - 1 && (3) \\ & \forall j, 1 \leq j \leq m : && (4) \\ & \quad \mathbf{S}^j \mathbf{p}_i \leq \mathbf{s}^j + M a_i^j \mathbf{1} && (5) \\ & \quad \mathbf{p}_i^T \mathbf{n}^j = e^j + \beta_i^j && (6) \\ & \quad ||\beta_i^j||_1 \leq M a_i^j && (7) \end{aligned}$$

with M being a sufficiently large constant[2] and $\mathbf{1}$ being a vector of appropriate size filled with ones. If $a_i^j = 0$, (7) implies than $\beta_i^j = 0$ and as a result (1) is satisfied and $\mathbf{p}_i \in \mathcal{S}_j$. If $a_i^j = 1$, then for a sufficiently large M, (5), (6), and (7) always have a solution and are effectively ignored. We define a cardinality function $\mathbf{card}(\cdot)$ that counts the number of non-zero entries in a vector. Therefore, (3) ensures that one contact surface is always selected.

The complete MIP formulation of our problem is thus:

$$\begin{aligned} \textbf{find} \quad & \mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{3\times n} \\ & \mathbf{A} = [\mathbf{a}_1, \cdots, \mathbf{a}_n] \in \{0,1\}^{n\times m} \\ & \boldsymbol{\beta} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_n] \in \mathbb{R}^{n\times m} \\ \textbf{min} \quad & l(\mathbf{P}) \\ \textbf{s.t.} \quad & \mathbf{P} \in \mathcal{I} \cap \mathcal{G} \cap \mathcal{F} \\ & \forall i, 1 \leq i \leq n : \\ & \mathbf{card}(\mathbf{a}_i) = m - 1 : \\ & \quad \forall j, 1 \leq j \leq m : \\ & \quad\quad \mathbf{S}_j \mathbf{p}_i \leq \mathbf{s}_j + M a_i^j \mathbf{1} \\ & \quad\quad \mathbf{p}_i^T \mathbf{n}^j = e^j + \beta_i^j \\ & \quad\quad ||\beta_i^j||_1 \leq M a_i^j\,. \end{aligned} \tag{8}$$

Assumptions

We make the following assumptions to guarantee that the MIP formulation is convex:

- $l(\mathbf{P})$ is a convex quadratic objective function.
- The contact surfaces are "quasi-flat" [29] and do not intersect with one another. This limitation is due to our dynamics constraint formulation, as explained in [26].
- Dynamic constraints are verified by computing a "quasi-static" trajectory for the center of mass (COM).
- Kinematics constraints on the COM are approximated as linear inequalities.
- The gait (i.e. the contact order for the effectors) is given.
- For all $\mathbf{p}_i$, the orientation around the $\mathbf{z}$ axis of the foot in contact is given. We show how the orientation can be computed automatically in Section V.

These assumptions define a convex feasible set $\mathcal{F}$.

V. Efficient Reduction of the Combinatorics

In (8), for each $\mathbf{p}_i$ we consider the complete set $\mathcal{S}$ of potential contact surfaces. However, it is reasonable to presume that for each $\mathbf{p}_i$ only a subset $\mathcal{S}_i \subset \mathcal{S}$ of contact surfaces are feasible, because of the geometric and dynamic constraints that bind each footstep with its neighbors. We approximate the subset $\mathcal{S}_i \subset \mathcal{S}$ by exploiting the reachability condition, as introduced in [27].

A. The reachability condition

Let us assume for now that when looking for a footstep position $\mathbf{p}_i$, we know the position and orientation of the root of the robot $\mathbf{R}_i \in SE(3)$ at the moment when the contact is created. In this case, a necessary condition for a surface $\mathcal{S}^j$ to be a valid candidate for $\mathbf{p}_i$ is that $\mathcal{S}^j$ be reachable with the foot from the root pose $\mathbf{R}_i$. Mathematically, we can define the discrete set of reachable surfaces for footstep i as:

$$\mathcal{S}_i = \{\mathcal{S}^j \in \mathcal{S} | ROM(\mathbf{R}_i, F_i) \cap S^j \neq \emptyset\} \tag{9}$$

where F_i is the foot of interest, and ROM is the 3D range of motion of the foot given a specific root pose $\mathbf{R}_i$ (Fig. 2).

[2]This formulation is known as the "Big M" formulation [30].

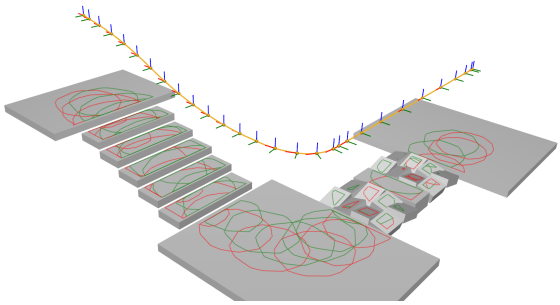

Fig. 3. Example of a planned root trajectory (yellow) with the intersected area between the ROM and the environment (red: left foot, green: right foot) at each discrete root pose. The displayed frames show the yaw ($\mathbf{z}$-axis) orientation of the trajectory at the discrete points, used as the yaw orientations of the contacting feet.

B. Kino-dynamic planning for estimating the root frames

In [28], we use the reachability condition to plan 6D trajectories for the root of the robot. Given the current frame $\mathbf{R}_0$ and a goal frame $\mathbf{R}_n$, we use a kino-dynamic RRT planner to compute a continuous root trajectory $\mathbf{R}(t), t \in [0, T]$ of the robot that always satisfies the reachability condition:

$$\forall t, \forall k, 1 \leq k \leq nFeet, ROM(\mathbf{R}(t), F_i^k) \cap S \neq \emptyset \qquad (10)$$

where each F_i^k denotes one of the $nFeet$ feet of the robot. Additionally, the trajectory prevents a collision for whole-body motions computed in its neighbourhood in practice.

We discretise $\mathbf{R}(t)$ with a time step δt, which is the only hyperparameter required in our framework. Assuming that a new step occurs every δt, we compute the number n of required footsteps to reach the target, as well as the estimated root poses $\mathbf{R} = [\mathbf{R}(\delta t), \mathbf{R}(2\delta t), \cdots, \mathbf{R}(T)]$ at each step creation. Fig. 3 shows an example of an $SE(3)$ root trajectory and its discretisation. The reachable space of each foot is shown for each step. We assume that for each $\mathbf{p}_i$, the orientation of the foot about the $\mathbf{z}$-axis is aligned with the root.

The kino-dynamic planner allows us to filter the contact candidates and pre-define the orientation of the feet, significantly reducing the number of surface candidates in our MIP. It is interesting to note that the information given by the kino-dynamic planner (number of steps and rotation about the $\mathbf{z}$-axis) is handled inside the quadratic cost in the MIP formulation by [19]. Their formulation has the advantage of being self-contained, although it does allow for consideration of non-flat contact surfaces. The choice of optimising the number of steps and orientation within the cost, as opposed to our potentially sub-optimal approach, deserves a discussion. We believe that the tuning required to weigh the several terms in the cost is as hard as the parametrisation of the δt parameter. The potential sub-optimality is the price to pay for the computational gains shown in our results, which are connected to the reduced combinatorics by our approach. We detail the kino-dynamic root trajectory planner in Appendix B.

VI. Convex Relaxation of the MIP Feasibility Problem

We purposely formulated (8) as a special instance of a *cardinality problem* [31], [32]. We observe that the constraint $\mathbf{card}(\mathbf{a}_i) = m - 1$ constrains $\mathbf{card}(\mathbf{a}_i)$ to its minimum: as all the $\mathcal{S}^j$ are disjoint, it is impossible for a footstep to lie simultaneously on more than one contact surface. Therefore, by removing the cardinality constraint and replacing the cost with $l(\mathbf{P}) + w \sum_{i=1}^{n} \mathbf{card}(\mathbf{a}_i)$, with w being a sufficiently large weight, we obtain a strictly equivalent problem.

Cardinality minimisation problems are well-known to be efficiently approximated with $\ell 1$-norm minimisation problems, thanks to the sparsity induced by the $\ell 1$-norm [33], [34]. Therefore, we replace the Boolean variables $\mathbf{a}_i$ with the real variables $\boldsymbol{\alpha}_i$ and minimise their norm to obtain what we call SL1M formulation [26]. Combining the $\ell 1$-norm relaxation with the pruning obtained through the use of the kino-dynamic planner as proposed in Section V, we obtain the following problem:

$$\begin{aligned}
\textbf{find} \quad & \mathbf{P} = [\mathbf{p}_1, \cdots, \mathbf{p}_n] \in \mathbb{R}^{3\times n} \\
& \boldsymbol{\alpha} = [\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_n], \boldsymbol{\alpha}_i \in \mathbb{R}^{+|S_i|} \\
& \boldsymbol{\beta} = [\boldsymbol{\beta}_1, \cdots, \boldsymbol{\beta}_n], \boldsymbol{\beta}_i \in \mathbb{R}^{|S_i|} \\
\textbf{min} \quad & l(\mathbf{P}) + w \sum_{i=1}^{n} \sum_{j=1}^{|S_i|} \alpha_i^j \\
\textbf{s.t.} \quad & \mathbf{P} \in \mathcal{I} \cap \mathcal{G} \cap \mathcal{F} \\
& \forall i, 1 \leq i \leq n : \\
& \quad \forall j, 1 \leq j \leq |\mathcal{S}_i| : \\
& \quad\quad \mathbf{S}_{i,j}\mathbf{p}_i \leq \mathbf{s}_{i,j} + M\alpha_i^j \mathbf{1} \\
& \quad\quad \mathbf{p}_i^T \mathbf{n}_{i,j} = e_{i,j} + \beta_i^j \\
& \quad\quad ||\beta_i^j||_1 \leq M\alpha_i^j
\end{aligned} \qquad (11)$$

where $|\mathcal{S}_i|$ is the number of elements of the set $\mathcal{S}_i$. $\mathbf{S}_{i,j}$, $\mathbf{s}_{i,j}$, $\mathbf{n}_{i,j}$ and $e_{i,j}$ are the constants defined for the j-th surface inside the set $\mathcal{S}_i$. We observe that since all the elements of $\boldsymbol{\alpha}$ are positive, $||\boldsymbol{\alpha}_i^j||_1 = \boldsymbol{\alpha}_i^j, \forall i, j$. We set $l(\mathbf{P}) = 0$ and $w = 1$ when solving a feasibility problem. This transforms the optimisation problem into a feasibility problem. Once the contact surfaces are fixed, the quadratic cost $l(\mathbf{P})$ is reintroduced to locally optimise the footstep positions on the selected surfaces.

For a solution of (11) to be a solution of (8), we need that $\mathbf{card}(\boldsymbol{\alpha}_i) = m-1, \forall i$. However, this might not be the case. To handle any potential non-sparse optimum, we use a brute-force approach. We fix all the variables $\boldsymbol{\alpha}_i$ for which the cardinality constraint is satisfied. We then test all the combinations for the remaining free variables until either i) a solution is found, ii) a maximum number of trials is reached, or iii) all possibilities are exhausted. This approach is not fail-proof, but this is a good compromise for our goal of finding a solution by solving a minimum number of optimisation problems.

An interesting observation is that because we are solving a cardinality problem, (11) is very close to the relaxed problem initially solved by a MIP solver for (8). It would be equivalent if the cardinality constraint was also reformulated as part of the cost and the pruning of contact surfaces was included in (8). In Section VII, we will show that because of the preliminary operations run by a MIP solver, it is more computationally efficient to directly solve (11) if the problem converges to a feasible solution.

A. Extension to quadruped locomotion

In terms of discrete variables, the proposed approach extends directly to any number of legs as long as the gait is pre-defined, leaving (11) and (8) unchanged. A conservative change must however be brought into the quasi-static constraints defining $\mathcal{F}$ to guarantee a convex formulation. Quasi-static bipedal locomotion requires the COM to lie above the foot when the other foot breaks contact. For a quadruped, this constraint is not as limiting, as the COM can lie anywhere in the convex hull of the current contact points. However, if both the COM and feet locations are variables, the convex hull constraint is not convex. To preserve convexity, we express the COM's x and y coordinates as linear functions of the positions of the effectors (i.e. feet) in contact:

$$\mathbf{c}_{i_{x,y}} = \sum_k w_i^k \mathbf{p}_{i_{x,y}}^k, \quad \sum_k w_i^k = 1, \quad w_i^k \geq 0 \quad \forall i \tag{12}$$

where $\mathbf{p}_i^k$ represents the position of the k-th effectors in i-th contact, and w_i^k is a user-defined unit weighting vector that is fixed for each contact. In our tests, we average the weights, writing $w_i^k = 1/size(w_i)$.

VII. Experimental Results

We tested our approach in simulation in a variety of environments for the humanoid Talos [35] and the quadruped ANYmal [36]. To experimentally validate our framework, we provide both quantitative and qualitative comparisons against the different approaches explained. These results highlight the performance improvements by reducing the combinatorial complexity of the footstep planning problem using a trajectory.

A. Implementation details

The test environment is written in Python. The actual initialisation and resolution of both (8) and (11) are written in C++ and invoked through dedicated Python bindings. All optimisation problems are solved using the Gurobi [37] solver through those bindings[3]. Open-source LP/QP solvers such as Quadprog [38] and GLPK [39] give similar performance results for the case of SL1M. However, this is not true for the MIP formulation. We chose Gurobi for a fair comparison. The measured computation times represent the time spent in the initialisation and resolution of the optimisation problem by the solver. The trajectory planner is implemented in C++ within the HPP framework [40] and invoked from Python using a CORBA architecture. Tests were run on a PC with an AMD Ryzen 7 1700X eight-core processor on Ubuntu 18.04.

B. Quantitative analysis

We selected four environments representative of the difficulty of a combinatorial footstep planning problem (Fig. 4). In each scenario, the objective is to find a sequence of footsteps resulting in a feasible whole-body motion containing the given initial and goal root poses. For each scenario, we solve the footstep planning problem using three methods: MIP optimisation, MIP feasibility, and SL1M. MIP optimisation corresponds to (8), where we set an objective function, just for comparison, that minimises the sum of the squared distances between the planned footsteps. MIP feasibility and SL1M represent (8) and (11) respectively with $l(\mathbf{P}) = 0$, solving a feasibility problem. Once the contact surfaces are fixed, we locally optimise the footstep positions on the selected surfaces. To achieve this, we call an instance of (8), where all integer variables are fixed and the quadratic cost $l(\mathbf{P})$ is reintroduced.

In addition, each method is tested twice with the full combinatorics and with reduced combinatorics based on the trajectory. The same number of footsteps n computed by discretising the trajectory is used in both cases.

Table II reports the computation times in milliseconds of each method averaged over 100 runs, together with the number of footsteps and the average number of candidate surfaces per contact. Empty cells mean that the method was not able to converge to a feasible solution. In all other cases, the success rate was 100% for all the methods.

1) Easy scenarios (bridge and stairs): First, we consider the environments composed of less than 10 candidate surfaces. The results show that integrating the trajectory always reduces the computation times of the optimisation problem, but the total computation time (including the guide path computation) is higher. SL1M always outperforms the other methods, being up to 7.1 times faster than MIP optimisation and up to 3.7 times faster than MIP feasibility.

2) Hard scenarios (rubbles and rubbles & stairs): We also consider environments composed of more than 10 candidate surfaces, including sloped surfaces. We decided to mark SL1M without trajectory planning as a failure (blank in Table II) as the remaining combinatorics required more than 4,000 trials (Section VI). Table II shows that SL1M with trajectory planning configuration always converges and is also always the most computationally efficient.

The pruning improved the performance by a factor of 146.8 in the best case in the stairs & rubbles scenario with the MIP feasibility formulation. SL1M always outperforms the other methods and is up to 8.8 times faster than MIP optimisation and up to 3.1 times faster than MIP feasibility.

Table III reports the optimal values obtained by both MIP and SL1M using a cost function that minimises the sum of squared distance traveled by the feet at each step. The MIP formulation always converges to the global optimum and thus provides a ground truth. SL1M first computes a feasible set of contact surfaces and then optimises the positions in a later stage. As expected, when MIP and SL1M select the same contact surfaces, the optimal cost is the same (bridge). Conversely, the cost differs when the contact surfaces selected differ, which is expected as SL1M is only computing a feasible solution. We also observe that the pruning based on the kino-dynamic planner always precludes the solver from finding the global optimum. However, we observe that the increase in the resulting cost is small (always less than 10%), which we consider to be an acceptable trade-off.

[3]The informed reader will observe that Python bindings already exist for Gurobi. However, the initialisation of a problem through Python is not computationally efficient, which justified the writing of a dedicated C++ library. This inefficiency also explains the difference in the computation times observed for the MIP resolution in the current work and in [26].

TABLE II
CONTACT PLANNING COMPUTATION PERFORMANCE EVALUATION

Scenario	# footstep	w/o trajectory planning				w/ trajectory planning				
		# surf.	MIP opt.	MIP feas.	SL1M	avg. # surf.	trajectory	MIP opt.	MIP feas.	SL1M
bridge	16	3	157.5	166.2	45.3	1.2	179.4	87.5	77.3	35.5
stairs	12	7	130.3	59.5	23.6	2.0	319.0	92.6	34.7	19.9
rubbles	16	18	1914.0	389.4	-	5.9	252.2	647.7	230.2	73.4
rubbles & stairs	32	24	98060.5	74710.5	-	3.9	501.8	1070.5	508.9	217.6

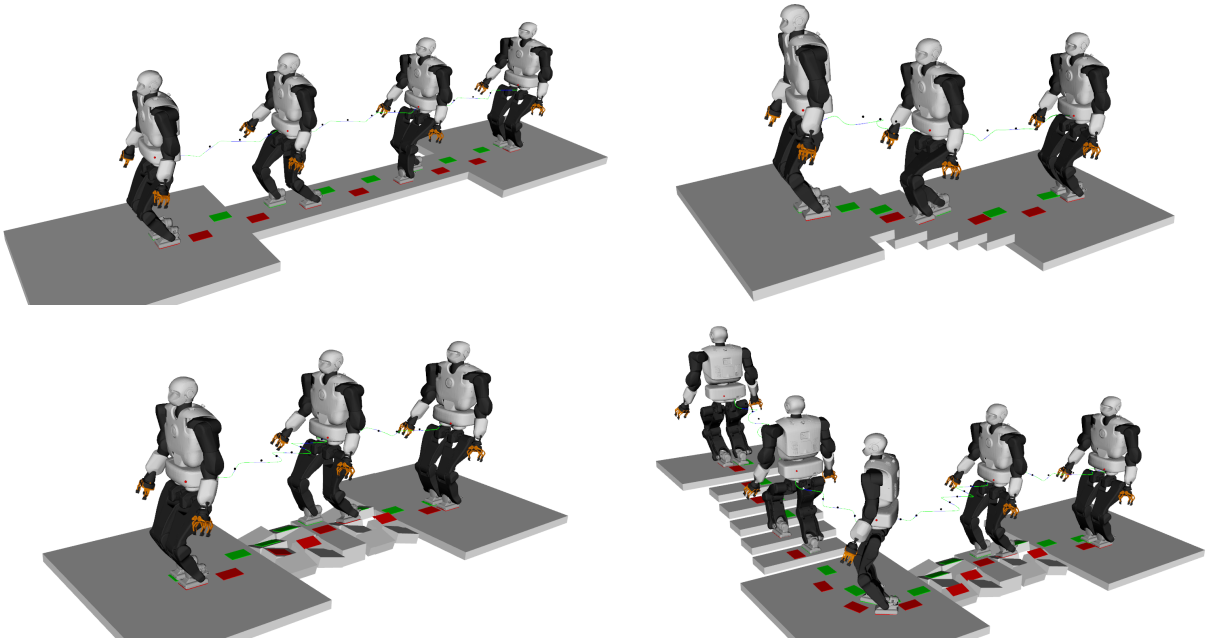

Fig. 4. Results of contact planning and the whole-body motion planning of Talos in our sample scenarios. The top row shows the *easy problems*, bridge and stairs. The bottom row shows the *challenging problems*, rubbles and rubbles & stairs.

TABLE III
COST FUNCTION VALUES

Scenario	w/o trajectory		w/ trajectory	
	MIP opt.	SL1M + QP	MIP opt.	SL1M + QP
bridge	1.887	1.887	1.974	1.974
stairs	0.797	0.872	0.870	0.929
rubbles	0.968	-	1.022	1.071
rubbles & stairs	1.880	-	2.072	2.224

Fig. 5. Contact planning results for ANYmal crossing the palet. The colored cubes indicate the planned contact for each foot.

C. Qualitative validation

We performed whole-body motion generation using [8], [41] to validate the computed footstep plans. Fig. 4 shows the snapshots of the generated motions for Talos along with the planned contact sequences. Although there are no theoretical guarantees that the contact plans can be extended to feasible whole-body motions (beside the quasi-static guarantee), this was the case in all of our scenarios. The companion video shows the resulting motions and compares them with the MIP formulation. The contact surfaces are selected using SL1M with trajectory planning, and then the footstep positions are optimised with a QP formulation. We also present a qualitative result obtained with the quadruped (Fig. 5).

D. Complete Pipeline

Table IV shows the average computation time in milliseconds, profiling the pipeline in SL1M with trajectory planner configuration. The computation times are broken down into i) trajectory planning, ii) contact surface selection with SL1M, and iii) QP-based footstep location optimisation.

VIII. DISCUSSION

From these experiments, we draw two conclusions. First, decreasing the possible combinations with a trajectory planner reduces the computation time needed to solve the tested optimisation problems. The improvement ranged from over

146.8 times to 1.180 times. It also improved the convergence of SL1M in challenging scenarios. Pruning the non-relevant candidates was more effective in challenging scenarios than in easy scenarios with a stronger reduction on the complexity ($\#surf^{\#footstep}$) of the problem. Indeed in easy scenarios, the time required for pruning outweighed the time savings in the optimisation phase. Second, solving feasibility problems (SL1M, MIP feasibility) is faster than solving optimisation problems (MIP optimisation) in general.

Always use the trajectory planner: Using a trajectory planner is not efficient for *easy problems* in terms of computation time. However, it brings other advantages such as automatic constraint construction and foot orientation information. Therefore, we recommend always using it. For *hard problems*, computational gains are remarkable. For instance, in the rubbles & stairs scenario, it is more than 100 times faster when compared with MIP feasibility without trajectory planning (yellow mark in Table II).

Non-sparse optimum handling: The $\ell 1$-norm relaxation can lead to a non-sparse optimum solution, where the relaxed integer variables do not converge to an integer value. A brute-force approach to handle this non-sparse optimum normally works well with easy problems. However, with challenging problems, we saw that it can result in large combinatorics. We tested specific methods to enhance the sparsity of $\ell 1$-norm minimisation, such as iterative re-weighting [42]. This did not improve the success rate for our problem, while in comparison, trajectory pruning was proved to be more efficient.

Combinatorial reduction in MIP problems: Before solving a combinatorial problem, MIP solvers commonly perform a "*presolve*" [24], where the problem is analysed, and unfeasible solutions are removed from the search space. Our trajectory planner can be considered as a domain-specific implementation of the presolve routine. Studying the influence of the presolve in challenging scenarios gives us some relevant insight into their combinatorics. The explored branch-and-bound node counts in MIP are shown in Table V. 0 means that MIP converged to integer values before getting into a branch-and-bound phase. When the trajectory is combined and the presolve is enabled, we can see that the MIP feasibility problems directly converge to a feasible solution. This is expected, as the first iteration of the MIP is equivalent to SL1M. More interestingly, this is also the case for the MIP optimisation problems, where 2960 nodes had to be explored to find the optimum without the trajectory. These results suggest that for our scenarios, even when considering a cost function, *there is no combinatorics involved in the resolution of the footstep planning problem*. Studying the validity of this hypothesis is an exciting direction for future work.

TABLE IV
COMPUTATION TIME FOR THE COMPLETE PIPELINE

Scenario	Trajectory	Footstep	Optimisation	Total
bridge	179.4	35.5	6.9	221.8
stairs	319.0	19.9	7.5	346.4
rubbles	252.2	73.4	8.0	333.6
rubbles & stairs	501.8	217.6	19.3	738.7

TABLE V
MIP EXPLORED NODE COUNTS

Scenario	Presolve	w/o trajectory		w/ trajectory	
		MIP opt.	MIP feas.	MIP opt.	MIP feas.
rubbles	disabled	2742	383	117	1
rubbles & stairs	disabled	20542	2724	7748	1
rubbles	enabled	1	1	0	0
rubbles & stairs	enabled	2960	1	0	0

Parameter tuning: Our framework requires the tuning of the discretisation step δt for the trajectory, as δt is used to infer the number of footsteps required by the motion (a new footstep is created every δt). If δt is too large, the problem can become unfeasible (not enough footsteps), while a small value increases the complexity of the problem (more footsteps and variables) and potentially leads to a failure in the convergence of SL1M. We heuristically selected a value $\delta t = 1.0s$ for our scenarios.We believe δt plays a central role in the performance of the framework and will investigate on automatically determining its value in future work.

SL1M with a cost function: Using SL1M jointly with an additional objective as in (11) is challenging. With the current state of our knowledge, we do not recommend this approach as it increases the risk of not converging to an integer solution. However, our results show that optimising the contact locations *after* selecting the contact surfaces leads to empirically convincing results, with a maximum increase of 10% of the optimal cost. Furthermore, considering the simplified models in footstep planning, the notion of optimality is loosely related to the optimality of the resulting whole-body motion. Therefore, the suboptimality may not be critical in practice.

IX. CONCLUSION

We presented a convex optimisation framework for planning the footsteps of a legged robot walking on uneven terrain. Our results suggest a positive answer to the question: *Can we efficiently address the combinatorics of the footstep planning problems with continuous optimisation methods?*

We showed that in our use-cases, the footstep planning problem can be relaxed as an $\ell 1$-norm minimisation problem (SL1M), converging to a feasible solution when the combinatorics involves less than 10 surface candidates per footstep. Problems with larger combinatorics can be pruned thanks to a kino-dynamic planner, which outputs an approximation of the trajectory followed by the root of the robot.

The benefits of the framework were measured in terms of the computational gains (more than 100 times faster than the original Mixed-Integer Program in the best case), as well as the simplicity of the approach, which can be implemented using off-the-shelf open-source numerical solvers at the cost of losing guarantees of optimality. Our analysis of the behaviour of MIP solvers further suggests that the combination of the kino-dynamic planner and pruning methods from the literature can potentially remove the combinatorics of the problem, even when an optimal solution is desired.

APPENDIX A
REACHABILITY CONSTRAINTS

We detail our kinematic and dynamic (here, static equilibrium) feasibility constraints, noted as a set $\mathcal{F}$ in our formulation. Two sets of constraints guarantee that the robot can follow the footstep plan without falling or violating joint limits. First, the position of the COM is constrained with respect to the contact points, following the 2-PAC approach [6]. This allows to continuously guarantee feasibility of the COM trajectory while only considering two COM positions at each contact phase. We recall the method for completeness and extend it to handle variable foot translations under the quasi-flat constraint. Second, the position of each effector is constrained with respect to the other effector in contact. This appendix is largely a reproduction of the constraints expressed in the original SL1M paper [26].

A. Center Of Mass constraints

To guarantee that a dynamically feasible trajectory exists for the i-th footstep, we use the 2-PAC formulation. We only need to choose 2 COM positions for each footstep, namely $\mathbf{c}_{i,0}$ and $\mathbf{c}_{i,1}$, to guarantee continuous feasibility. Let us define a *phase* as the time window between the moments two consecutive contacts are made.

1) Equilibrium constraints: For quasi-flat contact surfaces, a sufficient condition for the COM to allow for static equilibrium is: $\mathbf{c}_i \in \mathbf{conv}_i$ [29], where $\mathbf{conv}_i$ denotes the convex hull of all the contact points at phase i. For bipedal walking, this boils down to having the COM on top of the support foot. In this case $\mathbf{c}_{i,0}$ is constrained to lie above the support polygon of $\mathbf{p}_{i-1}$ (i.e., the support foot used in the transition from phase $i-1$ to i, which was the swing foot for phase $i-1$) at the beginning of phase i. We then constrain $\mathbf{c}_{i,1}$ to be above $\mathbf{p}_i$ at the end of phase i :

$$\begin{aligned} \mathbf{F}_{i-1}^j(\mathbf{c}_{i,0} - \mathbf{p}_{i-1}) &\leq \mathbf{f}_{i-1}^j + \mathbf{1}\alpha_{i-1}^j \\ \mathbf{F}_i^j(\mathbf{c}_{i,1} - \mathbf{p}_i) &\leq \mathbf{f}_i^j + \mathbf{1}\alpha_i^j \end{aligned} \tag{13}$$

where $\mathbf{F}_i^j$ and $\mathbf{f}_i^j$ are the matrix and vector defining the polygonal shape of the foot associated to phase i on surface $\mathcal{S}^j$. Note that these constraints depend only on the xy coordinates of the COM and the foot positions.

By convexity of the static equilibrium region, all points of the straight line segment $[c_{i,0}, c_{i,1}]$ satisfy the static equilibrium constraint. Similarly, the straight line segments $[c_{i-1,1}, c_{i,0}]$ and $[c_{i,1}, c_{i+1,0}]$ are also feasible because the COM stays above the support effector for all the duration of the single support phase.

2) Reachability constraints: We additionally constrain $\mathbf{c}_{i,0}$ and $\mathbf{c}_{i,1}$ to guarantee kinematic reachability. We stress that the kinematic constraints are only approximated here, thus the "guarantees" that we mention for feasibility are only valid for this simplified representation of the robot. The COM positions are linearly constrained as follows. First, for each effector we compute offline a polytope that approximates the reachable COM workspace: a large number of configurations of the robot are randomly sampled, and those who are collision-free and correspond to a "quasi-flat" pair of contacts are kept. For each of those configurations, the COM is expressed in the frame of a given effector. The convex hull of all the computed COM positions approximates the COM workspace in the effector frame. For each effector k, we thus obtain a 3D polytope ${}^k\mathcal{R}$: $\{\mathbf{c} \in \mathbb{R}^3, {}^k\mathbf{R}\mathbf{c} \leq {}^k\mathbf{r}\}$.

At contact phase i, for each contact surface $\mathcal{S}^j$ the orientation of the foot frame is constant. The yaw is given by the trajectory planner, while the roll and pitch are given by the surface orientation. We note $\mathcal{R}_i^j$ the rotated polytope associated with contact $\mathbf{p}_i$ at phase i, assuming it lies on surface $\mathcal{S}^j$. The translation is variable, thus the constraints depend linearly on the effector positions. Both COM positions $\mathbf{c}_{i,m}, m \in \{0,1\}$ at phase i are thus constrained by the two active contacts $\mathbf{p}_i$ and $\mathbf{p}_{i-1}$:

$$\mathbf{R}_l^j(\mathbf{c}_{i,m} - \mathbf{p}_l) \leq \mathbf{r}_l^j + \mathbf{1}\alpha_l^j \quad \forall j, \forall l \in \{i-1, i\}\,. \tag{14}$$

Here again, the slack variable $\boldsymbol{\alpha}$ is used such that only the constraints related to the selected contact surfaces are applied. By convexity of our (approximated) kinematic constraints, if they are satisfied for $\mathbf{c}_{i,0}$ and $\mathbf{c}_{i,1}$ for all i then they are continuously satisfied.

B. Relative foot position constraints

Similarly to the case of the COM reachability, we use a sampling-based approach to approximate the reachable workspace of each foot with respect to the others. For effector k, we obtain a polytope ${}^k\mathcal{Q}$: $\{\mathbf{p} \in \mathbb{R}^3, {}^k\mathbf{Q}\mathbf{p} \leq {}^k\mathbf{q}\}$ that constrains the other effector. If k is the moving effector at phase i on surface $\mathcal{S}^j$, we write the associated constraint set ${}^k\mathcal{Q}_i^j$. We then apply the same reasoning as for the COM to obtain the following constraints at each phase (omitting the k for clarity):

$$\mathbf{Q}_{i-1}^j(\mathbf{p}_i - \mathbf{p}_{i-1}) \leq \mathbf{q}_{i-1}^j + \mathbf{1}\alpha_{i-1}^j \quad \forall j\,. \tag{15}$$

APPENDIX B
KINO-DYNAMIC ROOT TRAJECTORY PLANNER

We detail the kino-dynamic root trajectory planner [28] used to efficiently reduce the combinatorics as proposed in Section V. The planner is implemented as a standard RRT algorithm with a specific steering method[4] to guarantee that every position of the root in the trajectory satisfies the reachability condition. Additionally the dynamic constraints of the system are approximated with a heuristic in order to obtain smoother trajectories for the root, empirically shown to be more easily extended to feasible motions.

Concretely the steering method augments the well-known Double Integrator Minimum Time (DIMT) method [43] to compute minimum time trajectories between two states of the robot considering the dynamic constraints applying to the Center Of Mass (COM).

[4]In a RRT planner, a graph of states is built by sampling random states and using a local 'steering method' to try to connect them. Linear interpolation is the most trivial option. Once the local trajectory has been computed, a validator checks whether the local trajectory satisfies the constraints of the system. If part of the trajectory satisfies the constraints, the last valid state is added to the graph and the partial trajectory is kept as the edge connecting the two states

Assuming that the COM position $\mathbf{c}$ is a linear function of the root position $\mathbf{R}$, given user-defined **symmetric** bounds on the COM dynamics along three orthogonal axis:

$$\begin{aligned} -\dot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \dot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \dot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \\ -\ddot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \ddot{\mathbf{c}}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} \leq \quad \ddot{\mathbf{c}}^{max}_{\{\mathbf{x},\mathbf{y},\mathbf{z}\}} . \end{aligned} \tag{16}$$

Given also an initial state $\mathbf{S}_0 =< \mathbf{R}_0, \mathbf{c}_0, \dot{\mathbf{c}}_0 >$ and a target state $\mathbf{S}_1 =< \mathbf{R}_1, \mathbf{c}_1, \dot{\mathbf{c}}_1 >$, the DIMT outputs a minimum time Bezier curve $\mathbf{c}(t)$ that connects exactly $< \mathbf{c}_0, \dot{\mathbf{c}}_0 >$ and $< \mathbf{c}_1, \dot{\mathbf{c}}_1 >$ and satisfies (16) without considering collision avoidance or the reachability condition. From $\mathbf{c}(t)$ we can directly retrieve the translation component of $\mathbf{R}(t)$, while the orientation is given by interpolating between $\mathbf{R}_0$ and $\mathbf{R}_1$.

However, the DIMT method is not directly applicable as for legged robots the center of mass acceleration bounds are neither constant nor symmetric, but state-dependent. The bounds correspond to the non-slipping condition, and are thus determined by the COM position, as well as the contact points and normals.

To address this issue, we propose a two-step method:

- We use the DIMT method with acceleration constraints computed for the initial state $\mathbf{S}_0$. The probable contact positions and normals are estimated using the reachability condition. Given the root position $\mathbf{R}_0$, we compute the set of obstacles intersected by the reachable workspace and arbitrarily choose one position on one of the available surfaces as the plausible contact. By doing so, we increase the odds that the trajectory $\mathbf{c}(t)$ be dynamically feasible in the neighborhood of $\mathbf{S}_0$, but not along the complete trajectory.
- Then, in the trajectory validation phase, we verify the dynamic equilibrium of the robot by estimating the probable contact points along the trajectory and verifying the Newton-Euler equation for the COM given this estimate. We also verify the reachability condition (which also approximates collision avoidance constraints). The returned trajectory $\mathbf{c}'(t)$ is the part of $\mathbf{c}(t)$ that satisfies all these constraints.

We refer the reader to the original paper [28] for details on how the acceleration bounds can be extracted from a state and estimated positions using the definition of the centroidal wrench cone and empirical evidence of the efficiency of this approach in spite of the heuristics introduced.

ACKNOWLEDGMENT

D. Song and Y.-J. Kim are supported in part by the ITRC/IITP program (IITP-2021-0-01460) and the NRF (2017R1A2B3012701) in South Korea. The other authors are supported by the H2020 project Memmo (ICT-780l684).

REFERENCES

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE Int. Conf. on Robotics and Automation*, vol. 2, 2003, pp. 1620–1626 vol.2.

[2] P. M. Wensing and D. E. Orin, "High-speed humanoid running through control with a 3d-slip model," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5134–5140.

[3] S. Caron, A. Escande, L. Lanari, and B. Mallein, "Capturability-based pattern generation for walking with variable height," *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 517–536, Apr. 2020. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01689331

[4] C. Brasseur, A. Sherikov, C. Collette, D. Dimitrov, and P. Wieber, "A robust linear mpc approach to online generation of 3d biped walking motion," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, 2015, pp. 595–601.

[5] J. Carpentier, R. Budhiraja, and N. Mansard, "Learning feasibility constraints for multicontact locomotion of legged robots," in *Proc. of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017.

[6] S. Tonneau, P. Fernbach, A. Del Prete, J. Pettré, and N. Mansard, "2pac: Two-point attractors for center of mass trajectories in multi-contact scenarios," *ACM Trans. on Graph. (TOG)*, vol. 37, no. 5, 2018.

[7] A. Escande, A. Kheddar, and S. Miossec, "Planning contact points for humanoid robots," *Robotics and Autonomous Systems*, vol. 61, no. 5, pp. 428–442, 2013.

[8] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios," *IEEE Transactions on Robotics*, pp. 1–16, 2020.

[9] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[10] M. Fallon, S. Kuindersma, S. Karumanchi, M. Antone, T. Schneider, H. Dai, C. P. D'Arpino, R. Deits, M. DiCicco, D. Fourie, *et al.*, "An architecture for online affordance-based perception and whole-body planning," *Joun. of Field Robotics*, vol. 32, no. 2, pp. 229–254, 2015.

[11] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Trans. on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.

[12] T. Bretl, "Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem," *The International Journal of Robotics Research*, vol. 25, pp. 317 – 342, 2006.

[13] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, "Motion planning for legged robots on varied terrain," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[14] X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne, "Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning," *ACM Trans. on Graph. (TOG)*, vol. 36, no. 4, 2017.

[15] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

[16] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Trans. Graph.*, vol. 31, no. 4, July 2012. [Online]. Available: https://doi.org/10.1145/2185520.2185539

[17] K. Yunt and C. Glocker, "Trajectory optimization of mechanical hybrid systems using sumt," in *9th IEEE International Workshop on Advanced Motion Control, 2006.* IEEE, 2006, pp. 665–671.

[18] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014. [Online]. Available: https://doi.org/10.1177/0278364913506757

[19] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[20] H. Dai, G. Izatt, and R. Tedrake, "Global inverse kinematics via mixed-integer convex optimization," *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019. [Online]. Available: https://doi.org/10.1177/0278364919846512

[21] D. Driess, O. Oguz, J.-S. Ha, and M. Toussaint, "Deep visual heuristics: Learning feasibility of mixed-integer programs for manipulation planning," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2020.

[22] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," 2020.

[23] B. Aceituno-Cabezas, C. Mastalli, H. Dai, M. Focchi, A. Radulescu, D. G. Caldwell, J. Cappelletto, J. C. Grieco, G. Fernández-López, and C. Semini, "Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2531–2538, 2017.

[24] Gurobi, "Mixed-integer programming (mip) – a primer on the basics," https://www.gurobi.com/resource/mip-basics.

[25] J. Wang, I. Chatzinikolaidis, C. Mastalli, W. Wolfslag, G. Xin, S. Tonneau, and S. Vijayakumar, "Automatic gait pattern selection for legged robots," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3990–3997.

[26] S. Tonneau, D. Song, P. Fernbach, N. Mansard, M. Taïx, and A. Del Prete, "Sl1m: Sparse l1-norm minimization for contact planning on uneven terrain," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[27] S. Tonneau, A. Del Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, "An efficient acyclic contact planner for multiped robots," *IEEE Transactions on Robotics*, vol. 34, no. 3, pp. 586–601, 2018.

[28] P. Fernbach, S. Tonneau, A. Del Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3701–3707.

[29] A. Del Prete, S. Tonneau, and N. Mansard, "Fast algorithms to test robust static equilibrium for legged robots," in *2016 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 1601–1607.

[30] J. Lofberg, "Big-m and convex hulls," http://cpc.cx/t2N, 2012.

[31] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM review*, vol. 38, no. 1, pp. 49–95, 1996.

[32] C. Lemaréchal and F. Oustry, "Semidefinite relaxations and lagrangian duality with application to combinatorial optimization," 1999.

[33] S. Boyd, "l1-norm methods for convex-cardinality problems," *Lecture notes*, p. 11. [Online]. Available: http://cpc.cx/sXO

[34] M. J. Abdi, "Cardinality optimization problems," Ph.D. dissertation, University of Birmingham, 2013.

[35] O. Stasse, T. Flayols, R. Budhiraja, K. Giraud-Esclasse, J. Carpentier, J. Mirabel, A. Del Prete, P. Souères, N. Mansard, F. Lamiraux, *et al.*, "Talos: A new humanoid research platform targeted for industrial applications," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*. IEEE, 2017, pp. 689–695.

[36] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 38–44.

[37] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2019. [Online]. Available: http://www.gurobi.com

[38] B. A. Turlach and A. Weingessel, *quadprog: Functions to solve Quadratic Programming Problems.*, 2011, r package version 1.5-4. [Online]. Available: http://CRAN.R-project.org/package=quadprog

[39] A. Makhorin, "Glpk," *http://www. gnu. org/s/glpk/glpk. html*, 2008.

[40] J. Mirabel, S. Tonneau, P. Fernbach, A.-K. Seppälä, M. Campana, N. Mansard, and F. Lamiraux, "Hpp: A new software for constrained motion planning," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 383–389.

[41] A. Del Prete, N. Mansard, O. Ramos, O. Stasse, and F. Nori, "Implementing torque control with high-ratio gear boxes and without joint-torque sensors," in *Int. Journal of Humanoid Robotics*, 2016.

[42] E. J. Candes, M. B. Wakin, and S. P. Boyd, "Enhancing sparsity by reweighted l1 minimization," *Journal of Fourier analysis and applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[43] T. Kröger, A. Tomiczek, and F. M. Wahl, "Towards on-line trajectory computation," in *IEEE/RSJ Inter. Conf. on Intelligent Robots and Systems (IROS)*, 2006.